Санкт-Петербургский государственный университет

Швыркова Александра Алексеевна

Выпускная квалификационная работа

Платформа автоматизации обработки медицинских данных

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2017 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
доц. к.т.н. Ю. В. Литвинов

Рецензент:
Директор отдела по репортингу отчетов
ООО «ТехЦентр Дойче Банка» Н. А. Левашко

Санкт-Петербург
2021

Saint Petersburg State University

Aleksandra Shvyrkova

Bachelor's Thesis

Platform for medical data processing automatization

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2017 "Software and Administration of Information Systems"*

Scientific supervisor:
C.Sc., assistant prof. Yurii Litvinov

Reviewer:
Director, «Deutsche Bank TechCentre» LLC
Nikolay Levashko

Saint Petersburg
2021

Оглавление

Введение **4**

1. Постановка задачи **6**

2. Обзор предметной области **7**

3. Описание решения **11**

4. Апробация платформы **21**

Заключение **23**

Список литературы **24**

Введение

Доступность медицинских данных и заключений по ним – это очень актуальная задача на сегодняшний день. Это происходит по нескольким причинам. Многие пациенты хотят узнать диагноз по различным снимкам или медицинским обследованиям. Хорошие специалисты могут находиться далеко, и не всегда есть возможность поговорить с ними лично. Врачи также иногда хотят узнать второе мнение по какому-либо медицинскому вопросу.

Так как технологии машинного обучения стремительно развиваются в последние годы, в том числе и в области медицины, и растёт точность моделей, то вместе с этим растёт и доверие людей к такому методу диагностирования заболеваний. Именно поэтому автоматизированные обработчики данных с встроенными методами машинного обучения являются перспективными инструментами для обнаружения патологий по медицинским данным.

На данный момент не было найдено общедоступного ресурса, решающего одновременно проблемы хранения, анонимизации и автоматической обработки медицинских данных, а также включающего в себя возможность обсуждения этих данных. Для решения упомянутых проблем было решено разработать платформу с простым интерфейсом, доступную любому человеку. Она также будет служить веб-фронтендом для фреймворка MIRF (Medical Images Research Framework) [20], который разрабатывается как кафедральный проект и содержит в себе обработчики медицинских данных. Сейчас пользоваться этими обработчиками могут только люди, имеющие навыки программирования, что сильно сужает целевую аудиторию и замедляет популяризацию автоматизированных методов диагностики. Разрабатываемая платформа позволит легко и удобно использовать возможности этого фреймворка. Задача разработки системы интересна тем, что медицинские данные надо будет корректно анонимизировать и хранить. Пользователи смогут запускать готовые алгоритмы и загружать в них свои данные для получения заключения в удобном формате. Помимо автоматизированных способов

получения диагноза, платформа будет иметь возможность задавать вопросы по медицинской тематике реальным врачам и пользователям.

1. Постановка задачи

Целью данной работы является разработка платформы для хранения и автоматизации обработки медицинских данных. Для достижения этой цели были поставлены следующие задачи:

- разработать архитектуру платформы;
- реализовать взаимодействие с фреймворком MIRF;
- изучить и применить существующие технологии анонимизации и хранения медицинских данных, в частности DICOM;
- реализовать контроллеры для взаимодействия с клиентом;
- реализовать клиентскую часть платформы;
- провести апробацию полученной платформы.

2. Обзор предметной области

1. **Обзор общедоступных платформ, работающих с медицинскими данными**

   - UpToDate [23] – веб-платформа с платными подписками, которая предлагает множество различных разделов для студентов, врачей и просто пациентов, где можно найти интерактивные варианты решения проблем и постановки диагнозов. Для удобства присутствует поиск по темам или ключевым словам. Из уникальных особенностей можно отметить огромное количество медицинских калькуляторов для различных областей медицины, таких как анестезиология или кардиология. Также присутствует возможность анонимной обратной связи с создателями и редакторами этого сервиса.
   - Medscape [11] – веб-платформа, где специалисты могут загружать в систему свои интересные клинические случаи. Затем по этим записям обычные пользователи могут искать то, что им требуется, используя фильтры по темам, форматам (текстовый, аудио, видео) или типам (статья в журнале, клинический случай и т.д.). На данном портале отсутствуют обсуждения и комментарии. Сайт служит как сборник различных медицинских статей и реальных клинических случаев.
   - RadioMed [17] – веб-платформа для пациентов, рентгенологов и смежных специалистов. Помимо функциональности, связанной с публикацией постов и их комментированием, есть возможность создавать и выполнять интерактивные задания, направленные на изучение новой медицинской информации.

   Все вышеописанные платформы объединяет то, что они не поддерживают хранение и загрузку медицинских файлов помимо картинок или фотографий, а также большинство служит как хранилище статических записей по медицинским темам, нежели как форумы для обсуждения.

Упомянутые сервисы имеют закрытый исходный код, в связи с этим были рассмотрены также инструменты с открытым исходным кодом для организации форумов, чтобы посмотреть технологии, которые они используют.

- Discourse [7] – инструмент для организации веб-форумов, серверная часть которого написана на Ruby. Поддерживает множество функций, которые включают в себя возможность создавать темы для обсуждений, каналы для тематического общения.
- Talkyard [21] – веб-форум, написанный на Scala. Основная функциональность совпадает с Discourse. Помимо этого присутсвуют оценки публикациям и комментариям, на основе которых меняется их отображение при просмотре. Также есть модерация контента.

В архитектуре этих систем можно выделить основные компоненты, такие как контроллеры для обработки http-запросов, DAO-классы для работы с базой данных, классы для аутентификации пользователей, а также множество вспомогательных классов, например, для работы с json-объектами и изображениями.

2. Особенности фреймворка MIRF

Автоматизированные обработчики данных состоят из последовательного набора шагов, каждый из которых является отдельным алгоритмом для преобразования данных. Такую последовательность из обработчиков называют конвейером.

MIRF является библиотекой для создания таких конвейеров для обработки медицинских данных. DICOM и NIfTI являются одними из наиболее используемых в современной медицине форматов хранения и визуализации изображений, полученных в ходе обследований. Именно поэтому основной фокус фреймворка направлен на поддержку этих форматов. Также присутствует поддержка ЭКГ-сигналов.

Архитектурный стиль этой библиотеки – Pipes&Filters, что означает, что конвейеры строятся из независимых блоков, которые обмениваются между собой данными.

Основными сущностями в MIRF являются:

- Data – абстрактный класс для хранения вспомогательной информации, связанной с обследованием или пациентом.
- Algorithm – интерфейс, который должны реализовывать классы-обработчики данных.
- PipelineBlock или AlgorithmHostBlock – классы, которые инкапсулируют в себе алгоритмы для преобразования данных. Они объединяются в конвейеры. Только наследники класса Data могут передаваться между блоками конвейера. Сами блоки не имеют доступа ни к каким данным кроме тех, что передаются напрямую в блок.

Уникальной особенностью данного фреймворка является возможность легко встраивать модели машинного обучения в качестве алгоритмов.

В 2020 году была добавлена микросервисная архитектура, которая позволяет разворачивать блоки на серверах независимо друг от друга, что позвоялет легко масштабировать приложение и параллельно исполнять трудоёмкие блоки-обработчики. Новая архитектура позволяет строить конвейер с помощью удобной json-конфигурации с информацией о блоках и связей между ними [2].

На Листинге 1 изображен пример конфигурации конвейера, который читает DICOM-изображения, накладывает на них маску и создает отчет с полученными изображениями. С помощью этой конфигурации строится ацикличный направленный граф, который представляет конвейер обработки данных, изображенный на Рис. 1.

```
[
  { ''id'': 0, ''blockType'' : ''ReadDicomImagesAlg'', ''children'': [1, 2] },
  { ''id'': 1, ''blockType'' : ''AddMaskAlg'', ''children'': [3] },
  { ''id'': 2, ''blockType'' : ''ConvertImagesToPdfAlg'', ''children'': [4] },
  { ''id'': 3, ''blockType'' : ''ConvertImagesToPdfAlg'', ''children'': [4] },
  { ''id'': 4, ''blockType'' : ''PrepareReportAlg'', ''children'': [5] },
  { ''id'': 5, ''blockType'' : ''SaveReportAlg'', ''children'': [] }
]
```

Листинг 1: Пример конфигурации конвейера.

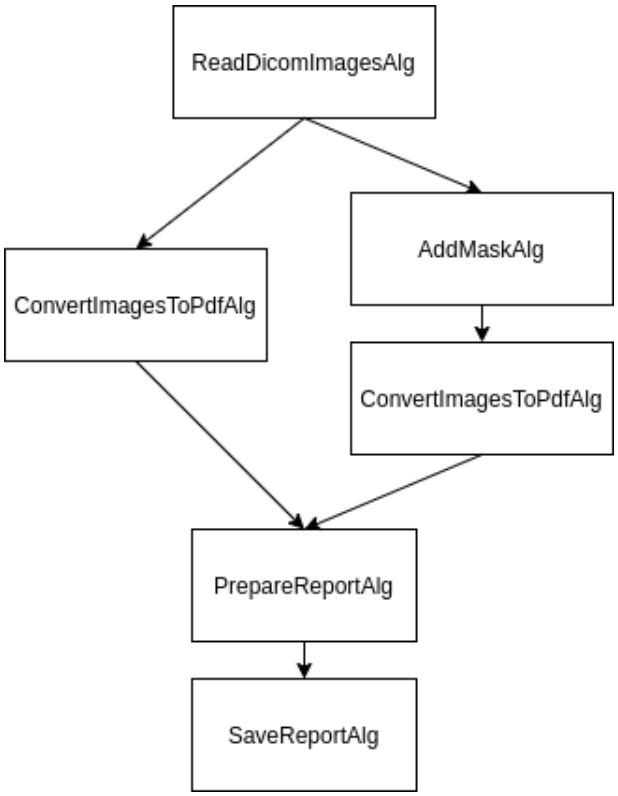

Рис. 1: Конвейер, полученный из json-конфигурации.

3. Описание решения

1. Архитектура

Было решено реализовать решение на языке Java с использованием библиотеки SpringBoot, так как этот способ разработки является одним из стандартов в области разработки веб-приложений, также имелся опыт работы с данным инструментов раннее. Этот способ разработки серверной части удобен тем, что уже содержит встроенный сервлет-контейнер Tomcat, который служит в роли веб-сервера и выполняет такие функции как первичная обработка входящих http-запросов или формирование ответа на запрос.

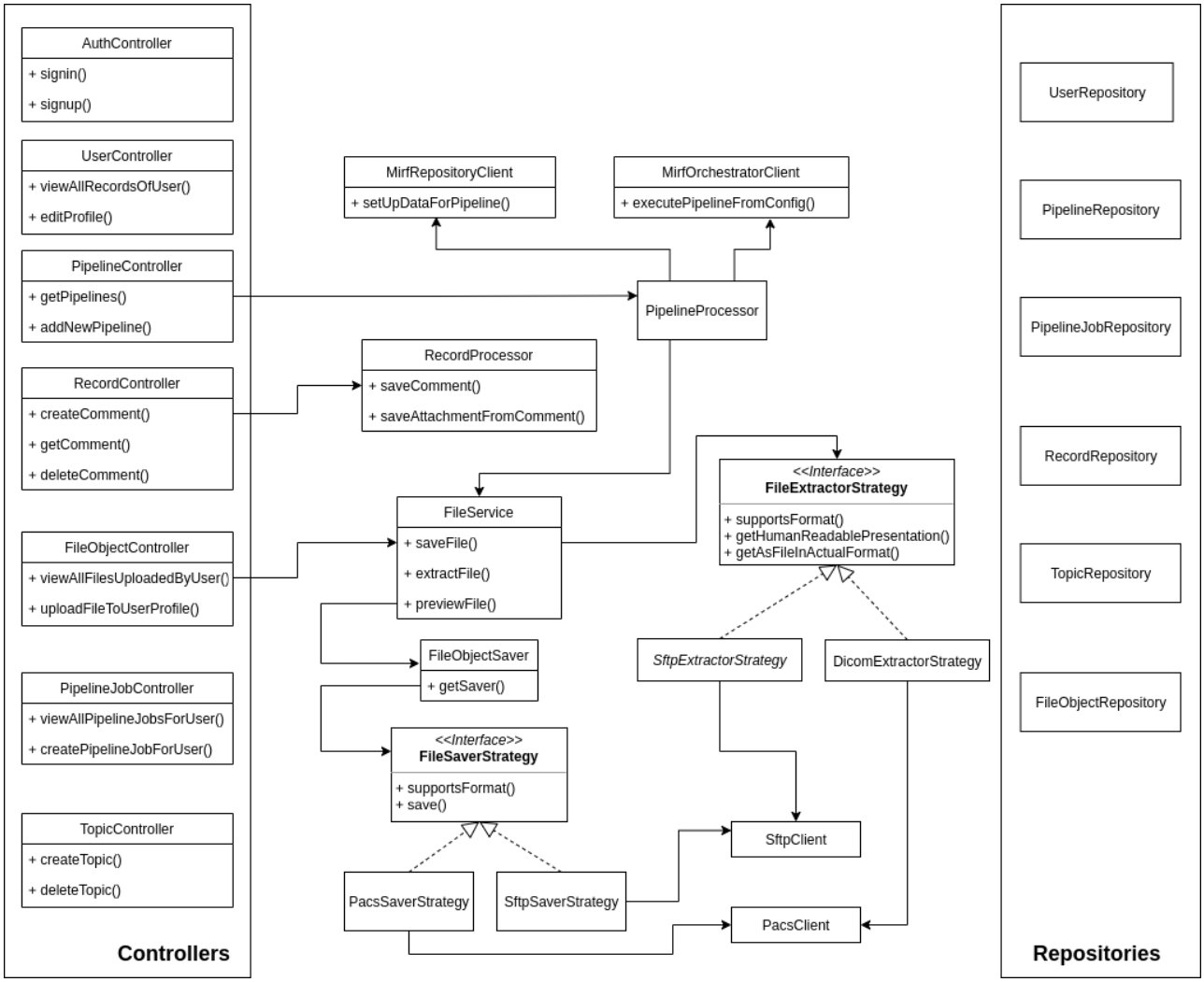

Рис. 2: Архитектура серверной части.

Диаграмма классов серверной части приложения представлена на Рис. 2. Она состоит из основных компонентов, присущих веб-приложениям. Контроллеры отвечают за обработку http-запросов, репозитории и модели используются для взаимодействия с базой данных. Основная логика приложения находится в классах-сервисах.

2. **Хранилище данных**

   Для данной работы необходима реляционная база данных с открытым исходным кодом. Самыми популярными БД такого типа являются PostgreSQL [14], MySQL [12] и MariaDB [9]. Выбор был сделан в пользу PostgreSQL, так как она поддерживает хранение массивов значений, индексируемый json-тип, частичные индексы, материализованные представления и другие особенности, которые не присутствуют в MySQL и MariaDB. Несмотря на то, что в небольших проектах MySQL используется гораздо чаще и имеет высокую скорость при операциях чтения, при возрастании нагрузки на проект PostgreSQL показывает результаты на различных операциях лучше, чем MySQL [15], [16].

   DICOM-изображения имеют сложную структуру, которая представляется в виде большого набора атрибутов и их значений. Чтобы корректно их хранить и обрабатывать, существует специальное решение в виде PACS-сервера (Picture Archiving and Communication System). Из решений с открытым исходным кодом можно выделить Orthanc [13], Dicoogle [6], EasyPACS [8]. Для интеграции с системой был выбран Orthanc-сервер, так как он содержит подробную документацию и разнообразное API для удобной работы с DICOM-изображениями. Был реализован класс OrthancClient с методами uploadInstance(), downloadInstance(), deleteInstance().

   Только базы данных может быть недостаточно, когда речь идет о файлах, имеющих большой размер, какими часто являются медицинские файлы, а также PDF-файлы. Результатом работы кон-

вейера обычно являются PDF-файлы. Их можно хранить в базе данных с типом BLOB или с использованием файловой системы. Исследование, проведенное в работе [19], показывает, что считывание и запись файла, чей размер превышает 1Мб, идёт эффективнее при использовании файловой системы. В связи с этим было решено использовать SFTP-сервер, который будет выступать в роли файловой системы с безопасным доступом по ssh. В базе данных будут храниться не сами файлы, а пути до них на файловой системе. Также SFTP-сервер сможет стать точкой расширения системы при добавлении новых типов медицинских данных, например, NIfTI-файлов[1], средний размер которых десятки мегабайтов.

Для взаимодействия с файловой системой был создан класс SftpClient, который устанавливает соединение с SFTP-сервером, имеет методы saveFile() и getFile() для загрузки и скачивания файлов.

Файлы, хранящиеся на PACS- и SFTP-серверах, можно будет затем прикреплять к записям с вопросами другим пользователям, а также использовать как входные данные для конвейеров.

3. **Взаимодействие с диагностирующей системой MIRF**

Для взаимодействия с MIRF в системе используются две сущности:

- Pipeline – хранит конфигурацию конвейера и метаинформацию, например, кто и когда создал данный конвейер.
- PipelineJob – представляет однократный запуск конкретного конвейера, а также хранит статус его исполнения.

Работа с MIRF организована в асинхронном стиле. Процесс выглядит следующим образом:

[1]NIfTI – стандарт хранения данных об изображениях головного мозга, полученных с помощью методов магнитно-резонансной томографии.

(a) Выбирается файл из уже загруженных в систему файлов. Выбранные входные данные для конвейера сериализуются и отправляются в формате архива в репозиторий-микросервис в MIRF.

(b) Посылается запрос на исполнение конвейера в оркестратор-микросервис (конвейеры, которые используют внутри себя нейронные сети, могут выполняться несколько минут, и чтобы не блокировать работу пользователя со всей системой, посылается запрос без ожидания результата).

(c) Если конвейер исполнился успешно, то оркестратор оповещает систему об успешном выполнении и возвращает отчёт. Иначе возвращается ошибка. Этот статус записывается в соответствующий экземпляр класса PipelineJob.

Также был немного модифицирован код фреймворка MIRF, чтобы иметь возможность собирать и запускать jar-файлы внутри docker-контейнеров. Для этого потребовалось добавить конфигурационные файлы и классы, а также сделать рефакторинг частей кода, которые до этого могли работать только локально.

Для демонстрации был развернут конвейер с классификацией внутричерепного кровоизлияния.

4. **Идентификация пользователей**

Для решения задачи авторизации и аутентификации пользователей используются Spring Security и Json Web Tokens (JWT).

Для аутентификации через API могут использоваться JWT-токены или sessionId в cookie-данных. После авторизации клиенту высылаются sessionId или токен соответственно. При последующих взаимодействиях со стороны клиента они передаются вместе с http-запросом на сервер, который проверяет их. Преимуществом JWT метода является то, что серверу не надо хранить дополнительную информацию в базе данных и, соответственно, выполнять лишние запросы к ней, в отличие от cookie-метода, для которого

требуется хранить все sessionId в БД. Также JWT-метод работает в микросервисной архитектуре, где аутентификация происходит между разными парами серверов.

JWT-токен состоит из трех частей: заголовок, полезная информация и подпись. Заголовок хранит алгоритм хэширования и тип токена. Полезная информация хранит поля, заданные пользователем, а также некоторые конфигурационные поля. Для генерации подписи используется секретный ключ, который хранится в серверной части приложения. С помощью этого ключа хэшируются данные из заголовка и полезных данных. Когда сервер получает от клиента запрос с токеном, он проверяет что подпись, генерируемая самим сервером с помощью ключа и хэширования, совпадает с подписью, полученной от клиента. Если подписи совпадают, то токен считается валидным, а пользователь – аутентифицированным.

Для решения данной задачи был реализован блок классов.

- OncePerRequestFilterImpl – наследует OncePerRequestFilter из фреймворка Spring, который гарантирует однократное применение фильтра по отношению к запросу. С помощью переопределенного метода doFilterInternal для каждого http-запроса валидируется его JWT-токен (если он присутствует) и сохраняет информацию об аутентификации пользователя в Spring Security Context.
- AuthenticationEntryPointImpl – реализует интерфейс AuthenticationEntryPoint из фреймворка Spring. Любой запрос от неаутентифицированного пользователя попадает сюда при попытке получить доступ к защищенным ресурсам.
- JwtUtils – класс для непосредственной работы с JWT-токенами. Методы generateJwtToken() и validateJwtToken() используются для генерации и проверки токенов соответственно.

5. **Клиентская часть**

   Frontend-часть системы написана с помощью JavaScript-фреймворка React. Также из использованных библиотек можно выделить:

   - Axios [3] – http-клиент для отправки запросов. Используется для связи с серверной частью приложения и получения данных.
   - Bootstrap [4], Material UI [10] – содержат различные css-шаблоны для оформления внешнего вида компонент.
   - React-validation [18] – проверяет корректное заполнение форм, например, при регистрации нового пользователя.

   Код состоит из трех смысловых частей: компоненты, сервисы и стили. В рамках данной работы были реализованы следующие компоненты и сервисы.

   - Регистрация и вход в систему.
   - Профиль пользователя, в котором он может загрузить файлы, а также посмотреть и скачать раннее сохраненные файлы.
   - Запуск обработчиков медицинских данных и просмотр результатов.
   - Создание постов с вопросами и возможностью комментирования.
   - Просмотр всех постов и поиск среди них по заголовку.

6. **Обезличивание данных на стороне клиента**

   Федеральный закон № 152-ФЗ определяет правила обработки персональных данных пользователей. Обработка и хранение таких данных подразумевает соблюдение различных требований, таких

как конфиденциальность данных, получение письменного согласия субъекта персональных данных и т.д. Анонимизация персональных данных упрощает эту задачу. Согласно ч. 1 ст. 7 Федерального закона № 152-ФЗ «обеспечение конфиденциальности персональных данных не требуется ... в случае обезличивания персональных данных»[2]

Именно поэтому была реализована процедура анонимизации данных, в частности DICOM-изображений. Обезличивание выполняется на клиентской стороне, таким образом информация передается на сервер и хранится уже в нужном виде. Чтобы распарсить DICOM-изображение и убрать все тэги, относящиеся к персональным данным, таким как фамилия, имя, дата рождения и др., была использована javascript-библиотека dcmjs [5].

7. **Развертывание приложения**

Для платформы был создан CI/CD пайплайн с помощью вебсервиса Travis[3]. Во время сборки приложения обновленные docker-образы сохраняются в облачном репозитории. Само приложение развертывается на Amazon Web Services[4] с помощью внутреннего сервиса Elastic Beanstalk.

Для того, чтобы не использовать командную строку при развертывании приложения и автоматизировать этот процесс, используется конфигурационный файл Dockerrun.aws.json, который содержит описание контейнеров. С помощью этого файла после успешной сборки будут выполнены инициализация и запуск новых контейнеров.

Также для удобного локального тестирования используется файл docker-compose.yml.

[2]О персональных данных: Федеральный закон от 27 июля 2006 г. № 152-ФЗ // Российская газета. – 2006. – Федеральный выпуск № 0 (4131). – Ст. 7.

[3]Travis CI. URL: https://www.travis-ci.com/

[4]Amazon Web Services. URL: https://aws.amazon.com/ru/

8. Внешний вид платформы

Интерфейс системы можно разделить на три основные части.

- Хранение файлов – пользователь может загрузить файлы со своего устройства (Рис. 3), а также просмотреть все загруженные раннее файлы и скачать их (Рис. 4).
- Автоматическая медицинская диагностика – пользователь может выбрать любой из доступных обработчиков медицинских снимков, а в качестве входных данных использовать раннее загруженные файлы (Рис. 5). Результаты диагностирования можно скачать и просмотреть в виде PDF-файла (Рис. 6).
- Форум – пользователь может посмотреть существующие посты других пользователей (Рис. 7). При полном просмотре поста с комментариями также есть предпросмотр медицинских снимков (Рис. 8).

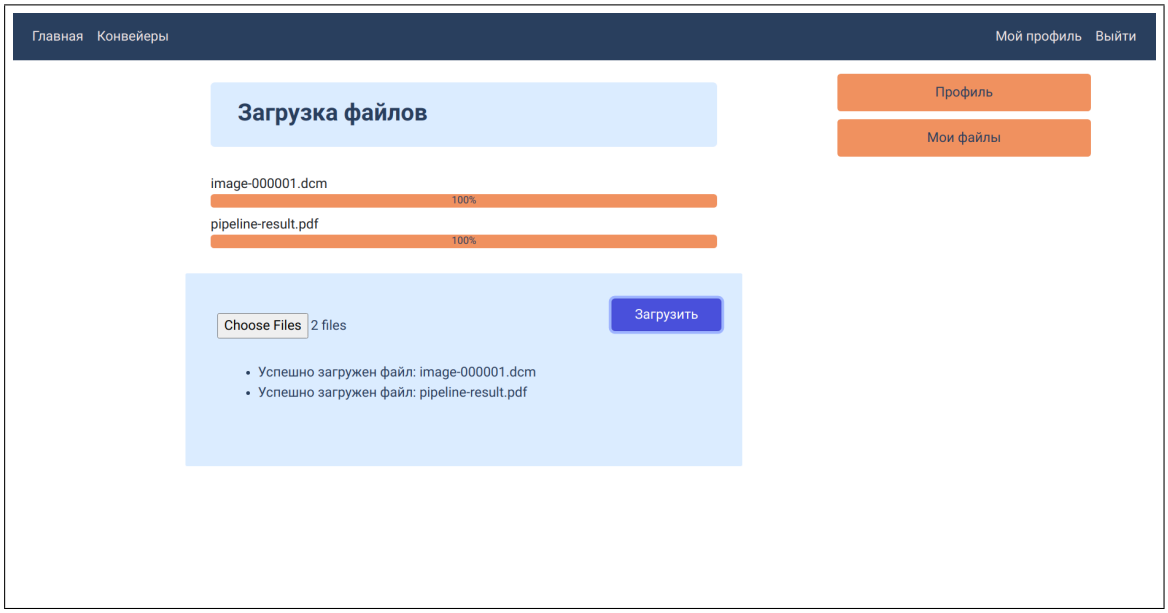

Рис. 3: Интерфейс загрузки файлов в профиль пользователя.

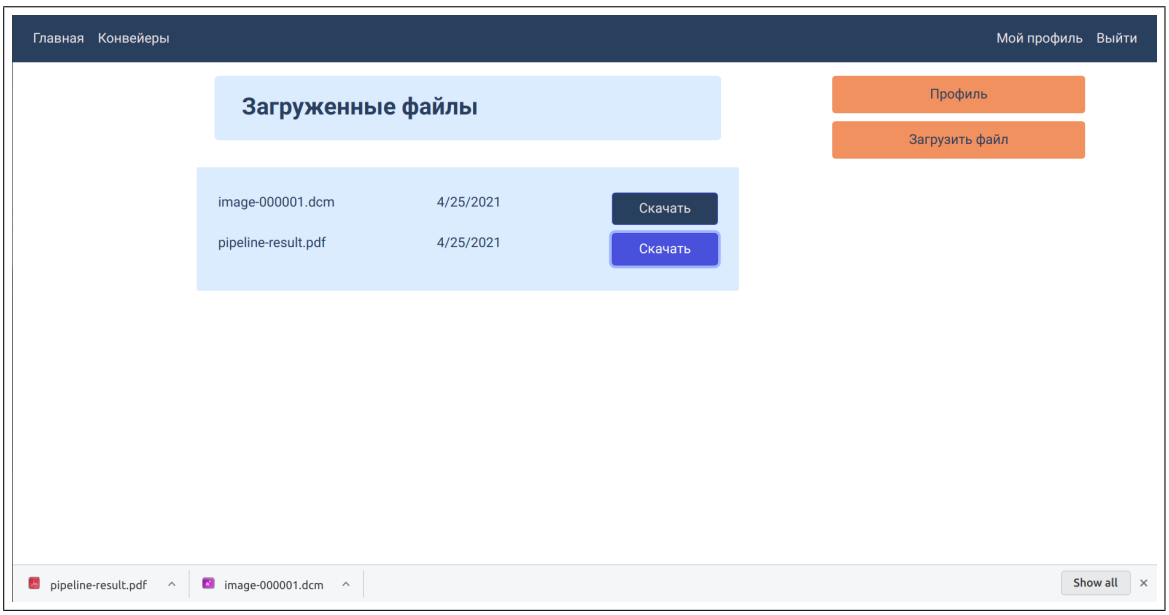

Рис. 4: Интерфейс просмотра и скачивания файлов, загруженных пользователем ранее.

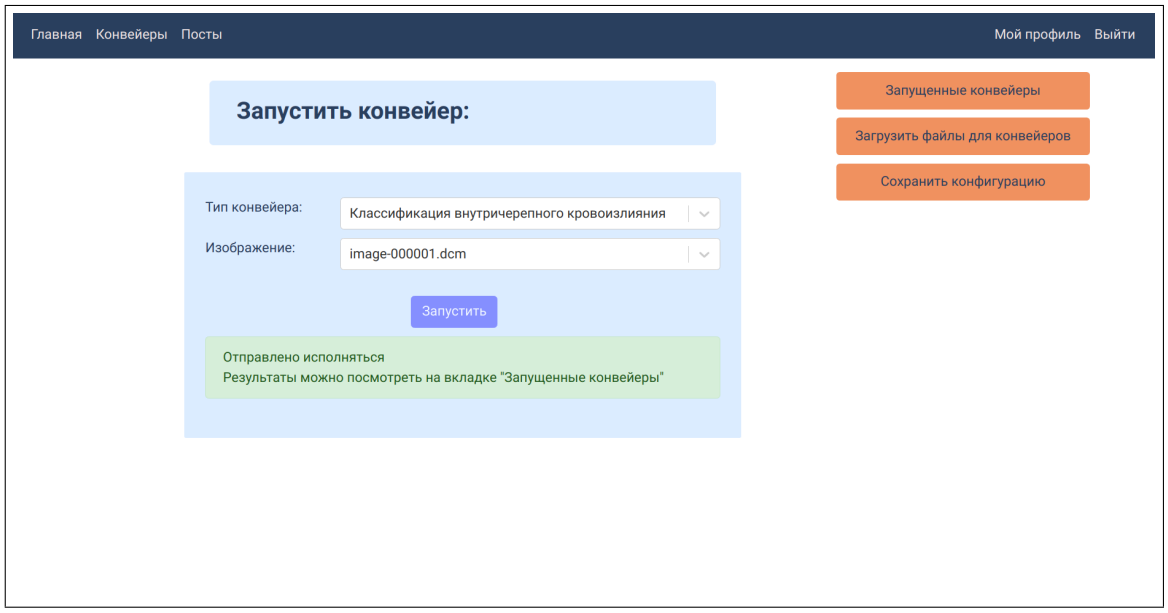

Рис. 5: Интерфейс запуска диагностики медицинских данных.

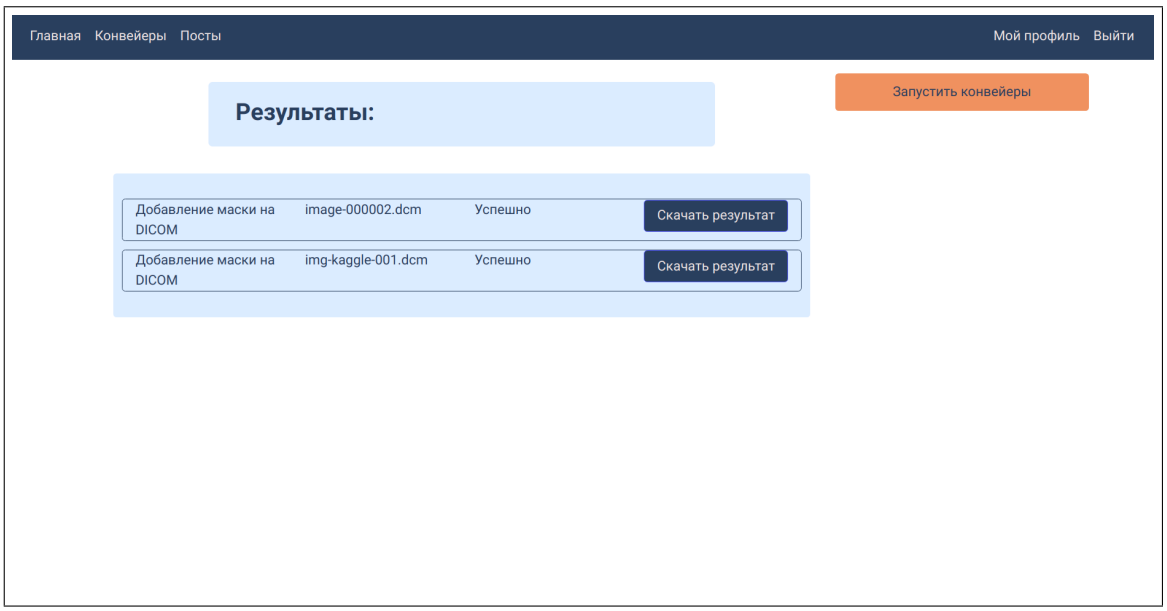

Рис. 6: Интерфейс просмотра результатов диагностирования.

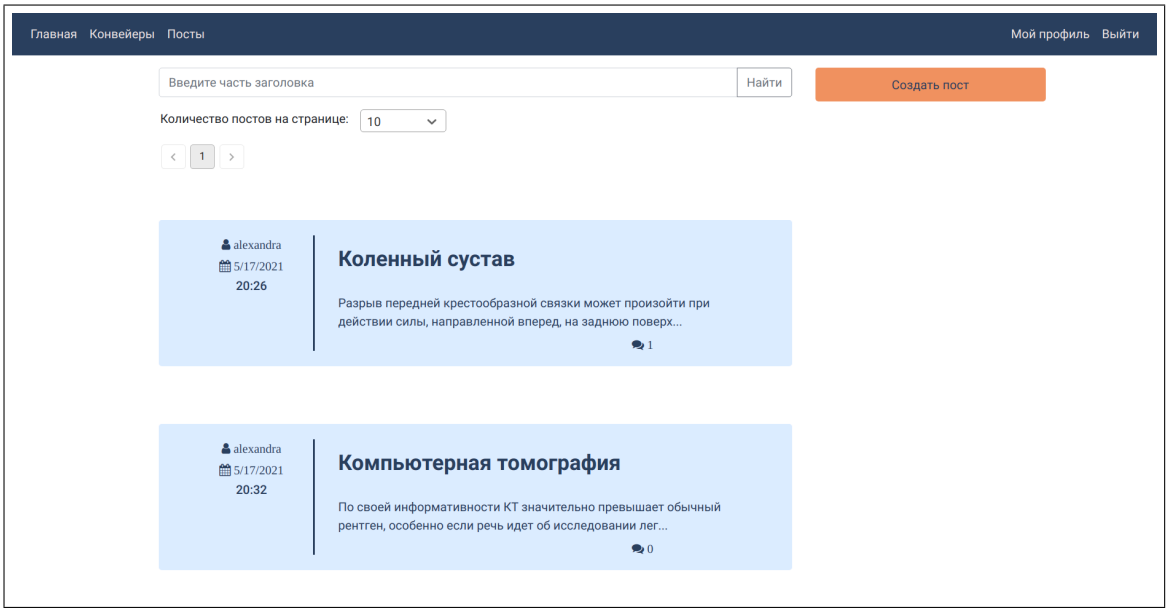

Рис. 7: Интерфейс просмотра всех опубликованных постов.

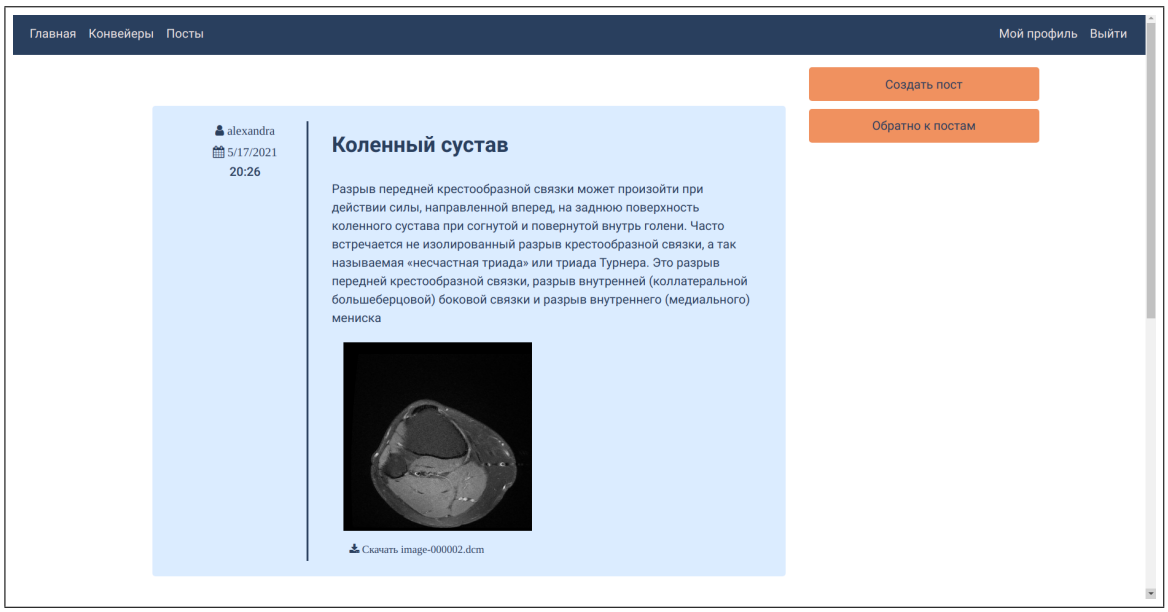

Рис. 8: Интерфейс просмотра поста полностью.

4. Апробация платформы

Основной целью данной работы являлось упростить использование конвейеров обработки данных из MIRF и сделать удобную и простую в использовании платформу для обычных пользователей, а именно врачей и пациентов. Удобство какой-либо системы или инструмента – это субъективный показатель, тем не менее, существует методы для оценки этого параметра.

При исследовании и сравнении различных методов [22] было уставлено, что System Usability Score является самым надежным, потому что имеет наименьшую разницу в результатах при большой и маленькой выборке опрашиваемых. Поэтому именно с помощью него будет оценена полученная платформа.

Суть метода состоит в том, чтобы по пятибальной шкале ответить на десять вопросов, связанных с использованием системы. Затем по определённой формуле высчитывается итоговое значение, которое варьируется от нуля до ста баллов.

Для сбора обратной связи была создана анкета. В тестировании платформы и последующем опросе были задействованы 10 чело-

век, включая врачей и обычных пользователей. Для каждого респондента было посчитано значение по методу System Usability Scale.

Результаты анкетирования представлены в Таб. 1.

Респондент	Total score	SUS score
1	28	70
2	36	90
3	37	92.5
4	36	90
5	30	75
6	32	80
7	36	90
8	34	85
9	34	85
10	33	82.5

Таблица 1: Результаты опроса

Среднее значение по всем опрошенным равно 84 балла. Однако это значение нельзя интерпретировать в качестве процентов. В работе [1] был проведен анализ более 3500 опросов по методу SUS. С помощью этого исследования была получена шкала нормализации итогового значения. Она позволяет дать четкую оценку исследуемой платформе. Всего выделено 8 оценок – от «Worst Imaginable» до «Best Imaginable». Значение 84 соответствует оценке «Good». А это значит, что платформа удобна в использовании, но всё еще нуждается в улучшениях.

При описании общего впечатления от системы респонденты также отметили простоту и доступность платформы для обычного пользователя. В качестве замечаний было упомянуто отсутствие описания конвейеров, а также было пожелание добавить более тонкие настройки в режим работы обработчиков данных.

Заключение

В ходе выполнения данной бакалаврской работы были получены следующие результаты:

- разработана архитектура веб-платформы;
- реализовано хранилище данных для различных типов файлов;
- реализовано взаимодействие с фреймворком MIRF;
- реализована аутентификация и авторизация пользователей;
- реализована клиентская часть платформы;
- произведена апробация платформы.

Код проекта: https://github.com/alexeevna/Medical-Web-App

Список литературы

[1] Aaron Bangor, Philip Kortum, James Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale // Journal of Usability studies 2009. – Vol.4 – Issue 3.

[2] Alexandra Shvyrkova, Alexey Fefelov, Yurii Litvinov, Angelina Chizhova, Egor Ponomarev, Alexander Lomakin, Alexander Savelev. MIRF 2.0 – A Framework for Distributed Medical Images Analysis // Software Engineering and Information Management 2020. – Vol.2691 – Issue 61.

[3] Axios documentation. – URL: https://github.com/axios/axios-docs (Дата обращения 25.04.21).

[4] Bootstrap documentation. – URL: https://getbootstrap.com/docs/4.1/getting-started/introduction/ (Дата обращения 25.04.21).

[5] Dcmjs. – URL: http://dcmjs.org/ (Дата обращения 01.05.21).

[6] Dicoogle documentation. – URL: https://dicoogle.com/ (Дата обращения 25.04.21).

[7] Discourse. — URL: https://www.discourse.org/(Дата обращения 18.10.20).

[8] EasyPACS. – URL: https://github.com/mehmetsen80/EasyPACS (Дата обращения 25.04.21).

[9] MariaDB documentation. – URL: https://mariadb.org/documentation/ (Дата обращения 14.12.20).

[10] MaterialUI documentation. – URL: https://material-ui.com/getting-started/usage/ (Дата обращения 10.05.21).

[11] MedScape. — URL: https://www.medscape.com/ (Дата обращения 18.10.20).

[12] MySQL documentation. – URL: https://dev.mysql.com/doc/ (Дата обращения 14.12.20).

[13] Orthanc documentation. – URL: https://book.orthanc-server.com/users.html (Дата обращения 25.04.21).

[14] PostgreSQL documentation. – URL: https://www.postgresql.org/docs/ (Дата обращения 14.12.20).

[15] PostgreSQL vs MySQL. – URL: https://www.2ndquadrant.com/en/postgresql/postgresql-vs-mysql/ (Дата обращения: 03.05.21).

[16] PostgreSQL vs. MySQL: A 360-degree Comparison. – URL: https://www.enterprisedb.com/blog/postgresql-vs-mysql-360-degree-comparison-syntax-performance-scalability-and-features (Дата обращения: 03.05.21).

[17] RadioMed. — URL: https://radiomed.ru/ (Дата обращения 18.10.20).

[18] React-validation documentation. – URL: https://openbase.com/js/react-validation/documentation (Дата обращения 25.04.21).

[19] Russell Sears, Catharine van Ingen, Jim Gray. To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem? – URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2006/04/tr-2006-45.pdf/ (Дата обращения 13.12.20).

[20] Sabrina Musatian, Alexander Lomakin Angelina Chizhova. Medical Images Research Framework. — 2019.

[21] TalkYard. — URL: https://www.talkyard.io/ (Дата обращения 18.10.20).

[22] Thomas Tullis, Jacqueline Stetson. A Comparison of Questionnaires for Assessing Website Usability // 2006.

[23] UpToDate. — URL: https://www.uptodate.com/home (Дата обращения 18.10.20).